Universidad Nacional Autónoma de México.

Facultad de ingeniería.

Profesor: Ing. Arturo Pérez de la Cruz.

Equipo 1.

Alumno	No. De cuenta
Chavez García Jesús Ángel	314233040
Hernández Hernández Pedro Daniel	314008767
Marín Barrera Jorge Jair	314110204

Grupo: 3

Computación gráfica e interacción humano-computadora.

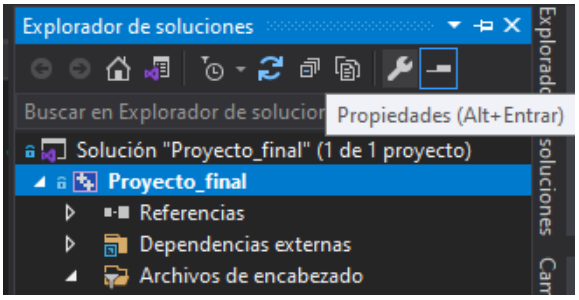
Manual técnico.

Índice

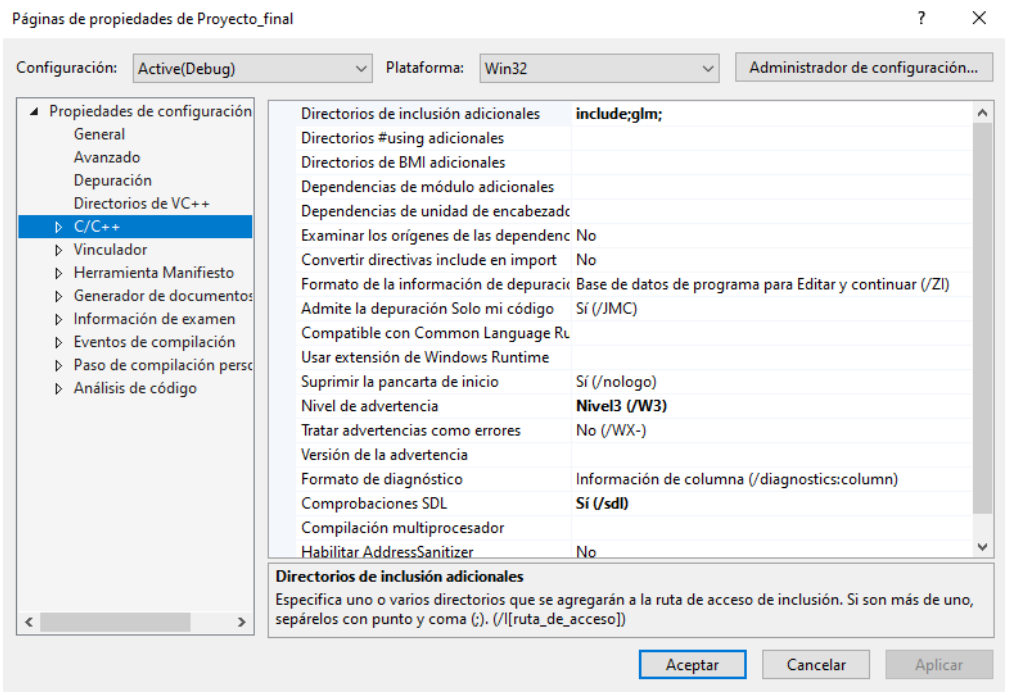
Configuración inicial

Para poder tener configurado de manera correcta el programa, se tendrán que seguir los siguientes pasos.

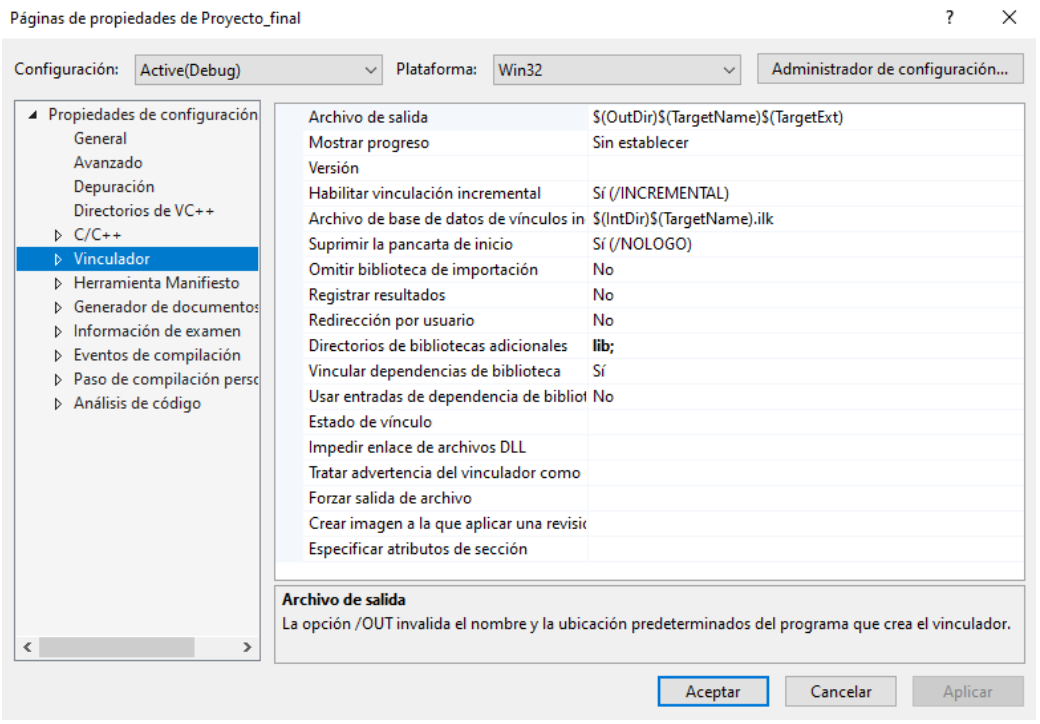
1. Tendremos que ir a las propiedades de nuestro proyecto.



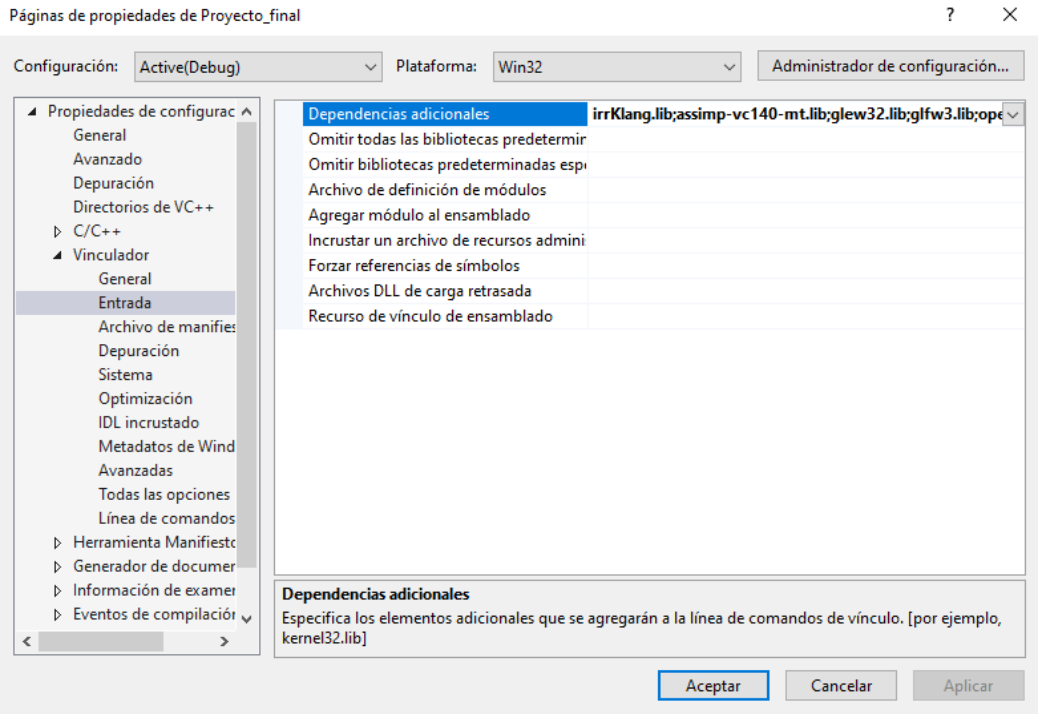
2. En las propiedades, deberemos de de ir a la sección llamada C/C++, en esta sección debemos de poner *include; glm;* en los *Directorios de inclusión adicionales*.



3. Posteriormente, deberemos de ir a la pestaña llamada *Vinculador*, específicamente la sección llamada General. Ahí deberemos de poner *lib;* en los *Directorios de bibliotecas adicionales.*



4. Finalmente, en la misma sección del vinculador, deberemos de poner en *Dependencias adicionales* lo siguiente: *irrKlang.lib;assimp-vc140-mt.lib;glew32.lib;glfw3.lib;opengl32.lib;* y, de este modo, ya tendremos configurado nuestro proyecto para poder ser usado



Cronograma

Proyecto. Animación de un acuario

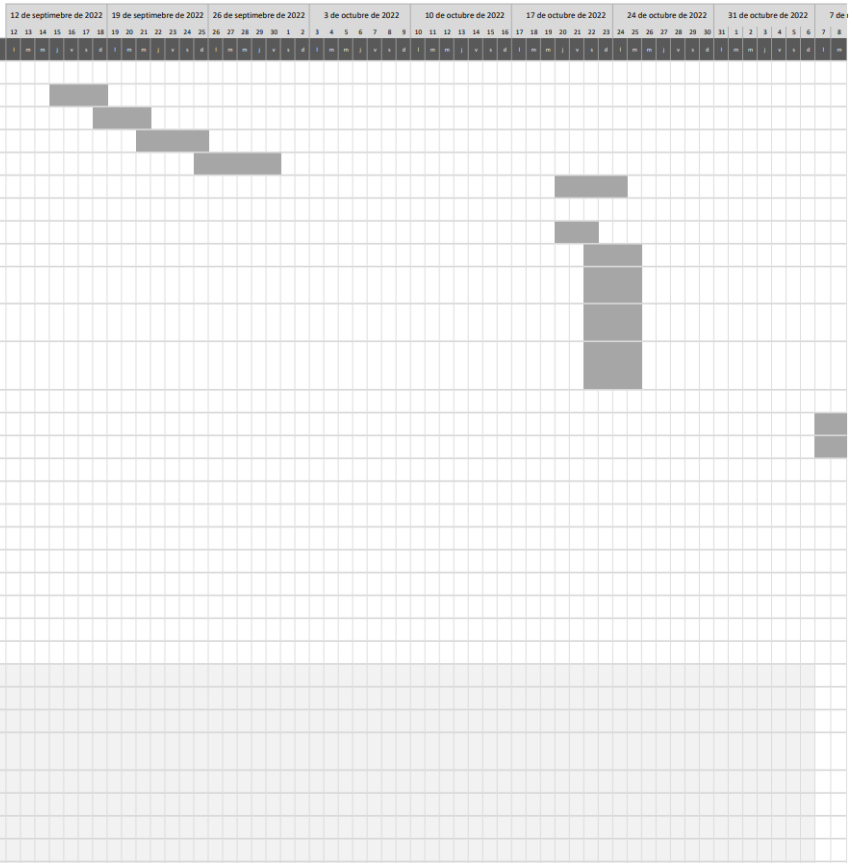
Equipo 01

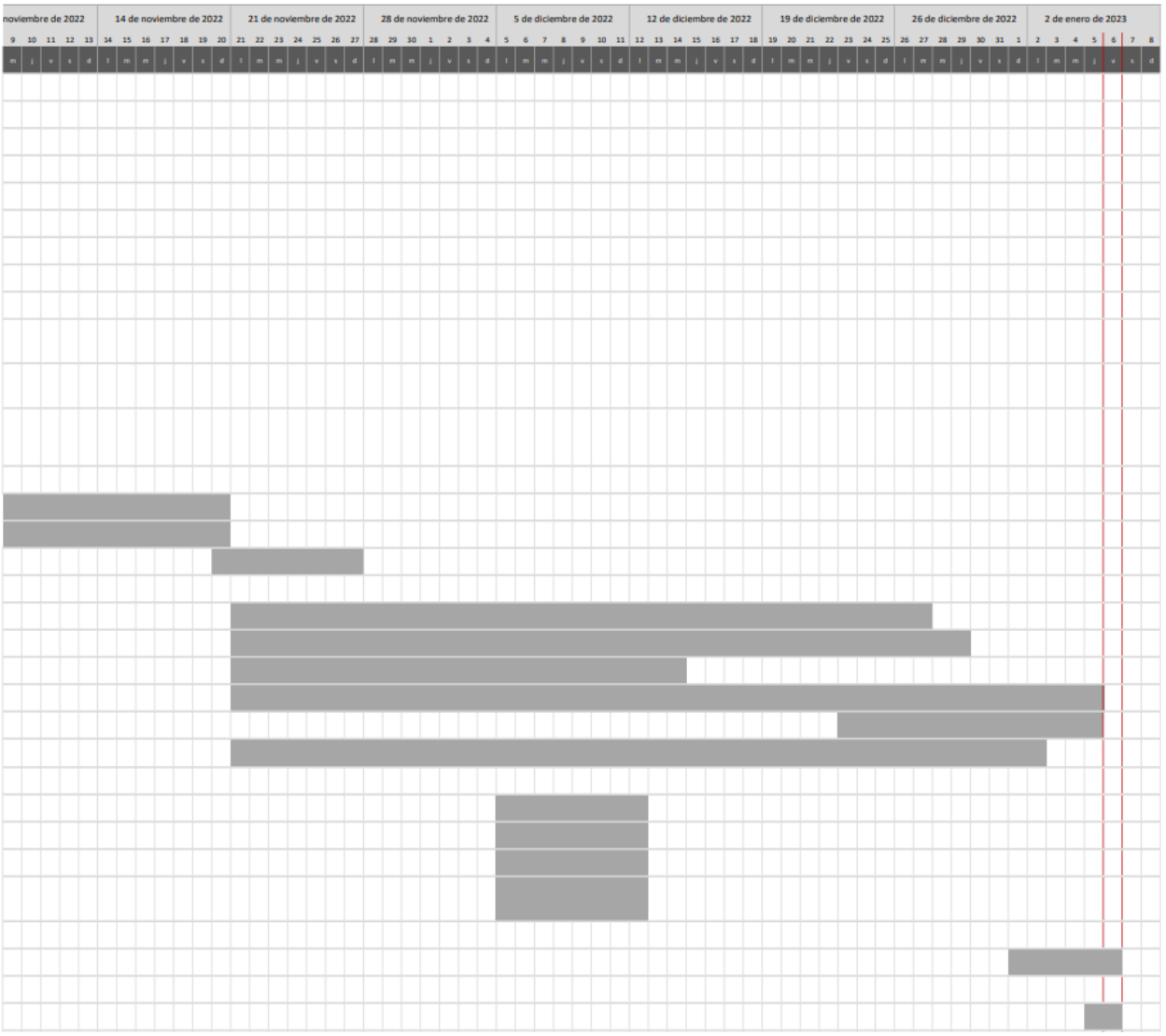
Integrante	No. de cuenta
-Chávez García Jesús Ángel	314233040
-Hernández Hernández Pedro Daniel	314008767
-Marín Barrera Jorge Jair	314110204

Inicio del proyecto: ju, 9/15/2022

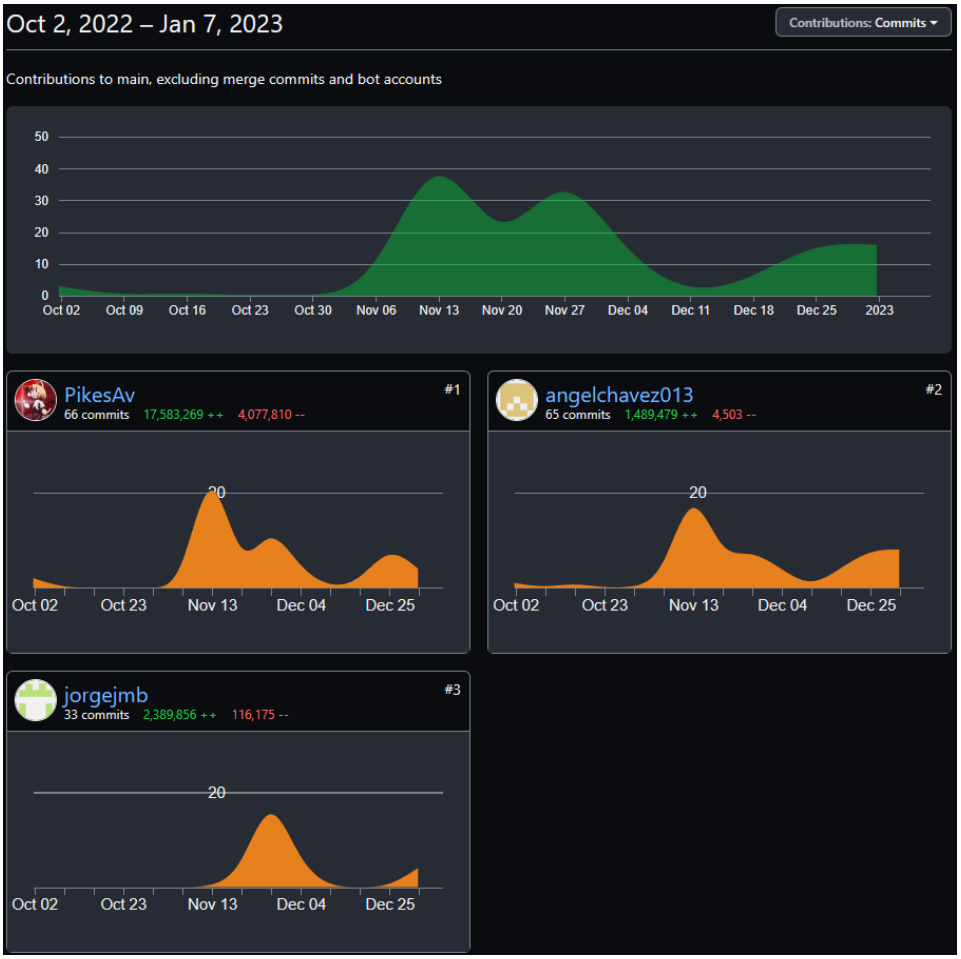
Semana para mostrar: 1

TAREA	ASIGNADO A	PROGRESO	INICIO	FIN
Creación del borrador del proyecto				
Busqueda de imágenes de referencia	Todo el equipo	100%	15-9-22	18-9-22
Propuestas de las animaciones	Todo el equipo	100%	18-9-22	21-9-22
Planeación del proyecto usando un diagrama de Gantt	Chávez García Jesús Ángel	100%	21-9-22	25-9-22
Precio del proyecto	Todo el equipo	100%	25-9-22	30-9-22
Busqueda de los modelos que posiblemente se vayan a usar	Todo el equipo	100%	20-10-22	24-10-22
Correción del borrador propuesto				
Correción de la carátula	Chávez García Jesús Ángel	100%	20-10-22	22-10-22
Elaboración del plano a diseñar	Hernández Hernández Pedro Daniel	100%	22-10-22	25-10-22
Colocación de los tiempos que se tendrán en la elaboración del proyecto	Chávez García Jesús Ángel	100%	22-10-22	25-10-22
Correción con respecto a las animaciones	Chávez García Jesús Ángel y Hernández Hernández Pedro Daniel	100%	22-10-22	25-10-22
Determinar el precio de venta del proyecto	Chávez García Jesús Ángel y Hernández Hernández Pedro Daniel	100%	22-10-22	25-10-22
Optimización y programación de los modelos a emplear				
Descarga de todos los modelos	Hernández Hernández Pedro Daniel y Marín Barrera Jorge Jair	100%	7-11-22	20-11-22
Optimización de los modelos a emplear	Hernández Hernández Pedro Daniel y Marín Barrera Jorge Jair	100%	7-11-22	20-11-22
Importación al código fuente	Hernández Hernández Pedro Daniel y Marín Barrera Jorge Jair	100%	20-11-22	27-11-22
Creación del código fuente del proyecto				
Elaboración y texturizado del acuario	Chávez García Jesús Ángel y Hernández Hernández Pedro Daniel	100%	21-11-22	27-12-22
Distribución de cada uno de los elementos del escenario	Todo el equipo, principalmente de Hernández Hernández Pedro Daniel	100%	21-11-22	29-12-22
Elaboración de cada una de las pesceras del acuario	Chávez García Jesús Ángel	100%	21-11-22	14-12-22
Elaboración de las animaciones	Marín Barrera Jorge Jair	100%	21-11-22	5-1-23
Audio	Chávez García Jesús Ángel	100%	23-12-22	5-1-23
Iluminación	Chávez García Jesús Ángel	100%	21-11-22	2-1-23
Implementación del proyecto de laboratorio				
Importación de todos los modelos	Hernandez Hernandez Pedro Daniel y Marín Barrera Jorge Jair	100%	5-12-22	12-12-22
Optimización y texturizado de los modelos a emplear	Hernandez Hernandez Pedro Daniel y Marín Barrera Jorge Jair	100%	5-12-22	12-12-22
Importación al código fuente	Hernandez Hernandez Pedro Daniel y Marín Barrera Jorge Jair	100%	5-12-22	12-12-22
Verificación de los posibles errores que se pudieran encontrar	Todo el equipo	100%	5-12-22	12-12-22
Pruebas finales del proyecto				
Busqueda y corrección de errores	Todo el equipo	100%	1-1-23	6-1-23
Entrega del proyecto				
Subida de todos los archivos faltantes al repositorio	Todo el equipo	100%	5-1-23	6-1-23





Vista general de la actividad en github


Oct 2, 2022 – Jan 7, 2023
Contributions: Commits
Contributions to main, excluding merge commits and bot accounts
50
40
30
20
10
0
Oct 02
Oct 09
Oct 16
Oct 23
Oct 30
Nov 06
Nov 13
Nov 20
Nov 27
Dec 04
Dec 11
Dec 18
Dec 25
2023
PikesAv
#1
66 commits
17,583,269 ++
4,077,810 --
20
Oct 02
Oct 23
Nov 13
Dec 04
Dec 25
angelchavez013
#2
65 commits
1,489,479 ++
4,503 --
20
Oct 02
Oct 23
Nov 13
Dec 04
Dec 25
jorgejmb
#3
33 commits
2,389,856 ++
116,175 --
20
Oct 02
Oct 23
Nov 13
Dec 04
Dec 25

Iluminación

La iluminación que se tiene tanto en la recepción, como en la zona del acuario, se pudo lograr con el siguiente código implementado.

```
//Luces correspondientes al festival de comida y de la recepcion del acuario
if (camera.getCameraPosition().y >= 0) {
    //Luces de la izquierda del festival
    if ((camera.getCameraPosition().x >= -140 && camera.getCameraPosition().x <= 0)
        && (camera.getCameraPosition().z <= 200 && camera.getCameraPosition().z >= -100)) {
        pointLights[0] = PointLight(1.0f, 1.0f, 1.0f,
            0.2f, 0.5f,
            -133.0f, 5.0f, -73.0f,
            0.2f, 0.001f, 0.0001f);
        pointLightCount++;
        pointLights[1] = PointLight(1.0f, 1.0f, 1.0f,
            0.2f, 0.5f,
            -133.0f, 5.0f, 5.0f,
            0.2f, 0.001f, 0.0001f);
        pointLightCount++;

        pointLights[2] = PointLight(1.0f, 1.0f, 1.0f,
            0.2f, 0.5f,
            -133.0f, 5.0f, 165.0f,
            0.2f, 0.001f, 0.0001f);
        pointLightCount++;
        Sonido = 2;
    }
    //Luces de la derecha del festival
    if ((camera.getCameraPosition().x >= 0 && camera.getCameraPosition().x <= 140)
        && (camera.getCameraPosition().z <= 200 && camera.getCameraPosition().z >= -100)) {
        pointLights[0] = PointLight(1.0f, 1.0f, 1.0f,
            0.2f, 0.5f,
            100.0f, 5.0f, -60.0f,
            0.2f, 0.001f, 0.0001f);
        pointLightCount++;
        pointLights[1] = PointLight(1.0f, 1.0f, 1.0f,
            0.2f, 0.5f,
            100.0f, 5.0f, 5.0f,
            0.2f, 0.001f, 0.0001f);
        pointLightCount++;
        pointLights[2] = PointLight(1.0f, 1.0f, 1.0f,
            0.2f, 0.5f,
            100.0f, 5.0f, 165.0f,
            0.2f, 0.001f, 0.0001f);
        pointLightCount++;
        Sonido = 2;
    }
```

```
    //Recepcion del acuario
    if ((camera.getCameraPosition().x >= -60 && camera.getCameraPosition().x <= 60)
        && (camera.getCameraPosition().z <= -150 && camera.getCameraPosition().z >= -259))
    {
        pointLights[0] = PointLight(0.85f, 0.85f, 0.85f,
            0.5f, 0.5f,
            -20.0f, 30.0f, -180.0f,
            0.2f, 0.01f, 0.0009f);
        pointLightCount++;
        pointLights[1] = PointLight(0.5f, 0.5f, 0.5f,
            0.5f, 0.5f,
            1.5f, 7.0f, -235.0f,
            0.2f, 0.01f, 0.0009f);
        pointLightCount++;
        pointLights[2] = PointLight(0.85f, 0.85f, 0.85f,
            0.5f, 0.5f,
            20.0f, 30.0f, -190.0f,
            0.2f, 0.01f, 0.0009f);
        pointLightCount++;
        Sonido = 1;
    }
    if ((camera.getCameraPosition().x >= -60 && camera.getCameraPosition().x <= 60)
        && (camera.getCameraPosition().z <= -140 && camera.getCameraPosition().z >= -150))
    {
        Sonido = 0;
    }
    //Escaleras
    if ((camera.getCameraPosition().x >= -60 && camera.getCameraPosition().x <= 60)
        && (camera.getCameraPosition().z <= -260 && camera.getCameraPosition().z >= -400))
    {
        pointLights[0] = PointLight(0.1f, 0.0f, 0.8f,
            0.2f, 0.5f,
            0.0f, 10.0f, -300.0f,
            0.2f, 0.01f, 0.0009f);
        pointLightCount++;
        pointLights[1] = PointLight(0.9f, 0.0f, 0.0f,
            0.2f, 0.5f,
            -20.0f, 10.0f, -375.0f,
            0.2f, 0.01f, 0.0009f);
        pointLightCount++;
        pointLights[2] = PointLight(0.5f, 0.2f, 0.8f,
            0.2f, 0.5f,
            20.0f, 10.0f, -375.0f,
            0.2f, 0.01f, 0.0009f);
        pointLightCount++;
```

```
3468                spotLights[0] = SpotLight(0.8f, 0.8f, 0.8f,
3469                    0.5f, 1.0f,
3470                    5.5f, 33.0f, -322.0f,
3471                    0.0f, -1.0f, 0.0f,
3472                    0.2f, 0.01f, 0.001f,
3473                    40.0f);
3474                spotLightCount++;
3475                shaderList[0].SetSpotLights(spotLights, 1);
3476                spotLights[0].SetFlash(glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3(0.0f, -1.0f, 0.0f));
3477                Sonido = 1;
3478            }
3479        }
```

```
3480        if (camera.getCameraPosition().y <= 0) {
3481            //Luces que se encuentran entre la pecera 1 y 2
3482            if ((camera.getCameraPosition().x >= 30 && camera.getCameraPosition().x <= 200)
3483                && (camera.getCameraPosition().z <= 200 && camera.getCameraPosition().z >= -400))
3484            {
3485                pointLights[0] = PointLight(0.4f, 0.6f, 1.0f,
3486                    0.2f, 0.5f,
3487                    90.0f, 5.0f, -340.0f,
3488                    0.17f, 0.001f, 0.0009f);
3489                pointLightCount++;
3490                pointLights[1] = PointLight(0.1f, 0.2f, 1.0f,
3491                    0.2f, 0.5f,
3492                    95.0f, 5.0f, -160.0f,
3493                    0.1f, 0.001f, 0.0009f);
3494                pointLightCount++;
3495                pointLights[2] = PointLight(0.4f, 0.6f, 1.0f,
3496                    0.2f, 0.5f,
3497                    0.0f, 5.0f, -150.0f,
3498                    0.17f, 0.001f, 0.0009f);
3499                pointLightCount++;
3500                Sonido = 3;
3501            }
3502            //Luces que se encuentran entre la pecera 2 y 3
3503            if ((camera.getCameraPosition().x >= -60 && camera.getCameraPosition().x <= 60)
3504                && (camera.getCameraPosition().z <= -120 && camera.getCameraPosition().z >= -300))
3505            {
3506                pointLights[0] = PointLight(0.4f, 0.6f, 1.0f,
3507                    0.2f, 0.5f,
3508                    -15.0f, 5.0f, -150.0f,
3509                    0.17f, 0.001f, 0.0009f);
3510                pointLightCount++;
3511                pointLights[1] = PointLight(0.1f, 0.2f, 1.0f,
3512                    0.2f, 0.5f,
3513                    -15.0f, 5.0f, -250.0f,
3514                    0.17f, 0.001f, 0.0009f);
3515                pointLightCount++;
3516                pointLights[2] = PointLight(0.0f, 0.3f, 0.9f,
3517                    0.2f, 0.5f,
3518                    -15.0f, 5.0f, -70.0f,
3519                    0.17f, 0.001f, 0.0009f);
3520                pointLightCount++;
3521            }
```

```
3522            //Luces que se encuentran al final del recorrido
3523            if ((camera.getCameraPosition().x >= -140 && camera.getCameraPosition().x <= -60)
3524                && (camera.getCameraPosition().z <= -120 && camera.getCameraPosition().z >= -400))
3525            {
3526                pointLights[0] = PointLight(0.4f, 0.6f, 1.0f,
3527                    0.2f, 0.5f,
3528                    -130.0f, 5.0f, -350.0f,
3529                    0.17f, 0.001f, 0.0009f);
3530                pointLightCount++;
3531                pointLights[1] = PointLight(0.1f, 0.2f, 1.0f,
3532                    0.2f, 0.5f,
3533                    -130.0f, 5.0f, -250.0f,
3534                    0.17f, 0.001f, 0.0009f);
3535                pointLightCount++;
3536                pointLights[2] = PointLight(0.4f, 0.6f, 1.0f,
3537                    0.2f, 0.5f,
3538                    -130.0f, 5.0f, -150.0f,
3539                    0.17f, 0.01f, 0.0009f);
3540                pointLightCount++;
3541            }
3542            //Luces que se encuentran en direccion a la pecera mas grande
3543            if ((camera.getCameraPosition().x >= 0 && camera.getCameraPosition().x <= 200)
3544                && (camera.getCameraPosition().z <= 200 && camera.getCameraPosition().z >= -120))
3545            {
3546                pointLights[0] = PointLight(0.4f, 0.6f, 1.0f,
3547                    0.2f, 0.5f,
3548                    110.0f, 5.0f, -75.0f,
3549                    0.17f, 0.001f, 0.0009f);
3550                pointLightCount++;
3551                pointLights[1] = PointLight(0.0f, 0.3f, 0.9f,
3552                    0.2f, 0.5f,
3553                    -15.0f, 5.0f, -70.0f,
3554                    0.17f, 0.001f, 0.0009f);
3555                pointLightCount++;
3556                pointLights[2] = PointLight(0.4f, 0.6f, 1.0f,
3557                    0.5f, 0.5f,
3558                    45.0f, 5.0f, 100.0f,
3559                    0.17f, 0.001f, 0.0009f);
3560                pointLightCount++;
3561            }
```

```
3562            //Luces que se encuentran en direccion a los pinguinos
3563            if ((camera.getCameraPosition().x >= -200 && camera.getCameraPosition().x <= 0)
3564                && (camera.getCameraPosition().z <= 200 && camera.getCameraPosition().z >= -120))
3565            {
3566                pointLights[0] = PointLight(0.4f, 0.6f, 1.0f,
3567                    0.2f, 0.5f,
3568                    -90.0f, 5.0f, -70.0f,
3569                    0.17f, 0.01f, 0.0009f);
3570                pointLightCount++;
3571                pointLights[1] = PointLight(0.0f, 0.3f, 0.9f,
3572                    0.5f, 0.5f,
3573                    -15.0f, 5.0f, -100.0f,
3574                    0.17f, 0.01f, 0.0009f);
3575                pointLightCount++;
3576                pointLights[2] = PointLight(0.4f, 0.6f, 1.0f,
3577                    0.5f, 0.5f,
3578                    -90.0f, 5.0f, 100.0f,
3579                    0.17f, 0.01f, 0.0009f);
3580                pointLightCount++;
3581            }
3582            //Luces que se encuentran entre los pinguinos y el recorrido tubular
3583            if ((camera.getCameraPosition().x >= -200 && camera.getCameraPosition().x <= 0)
3584                && (camera.getCameraPosition().z <= 200 && camera.getCameraPosition().z >= -70))
3585            {
3586                pointLights[0] = PointLight(0.4f, 0.6f, 1.0f,
3587                    0.2f, 0.5f,
3588                    60.0f, 5.0f, 100.0f,
3589                    0.17f, 0.01f, 0.0009f);
3590                pointLightCount++;
3591                pointLights[1] = PointLight(0.1f, 0.2f, 1.0f,
3592                    0.5f, 0.5f,
3593                    -5.0f, 5.0f, 30.0f,
3594                    0.17f, 0.01f, 0.0009f);
3595                pointLightCount++;
3596                pointLights[2] = PointLight(0.4f, 0.6f, 1.0f,
3597                    0.5f, 0.5f,
3598                    -90.0f, 5.0f, 100.0f,
3599                    0.17f, 0.01f, 0.0009f);
3600                pointLightCount++;
3601                spotLights[0] = SpotLight(1.0f, 1.0f, 1.0f,
3602                    0.5f, 1.0f,
3603                    -140.0f, 1.0f, 22.0f,
3604                    0.0f, -1.0f, 0.0f,
3605                    0.2f, 0.01f, 0.001f,
3606                    40.0f);
3607                spotLightCount++;
3608                shaderList[0].SetSpotLights(spotLights, 1);
3609                spotLights[0].SetFlash(glm::vec3(-130.0f, -10.0f, 30.0f), glm::vec3(0.0f, -1.0f, 0.0f));
3610            }
```

```
3611            //Luces que se encuentran entre la pecera 5 y el recorrido tubular
3612            if ((camera.getCameraPosition().x >= 0 && camera.getCameraPosition().x <= 200)
3613                && (camera.getCameraPosition().z <= 200 && camera.getCameraPosition().z >= -70))
3614            {
3615                pointLights[0] = PointLight(0.4f, 0.6f, 1.0f,
3616                    0.2f, 0.5f,
3617                    45.0f, 5.0f, 100.0f,
3618                    0.17f, 0.01f, 0.0009f);
3619                pointLightCount++;
3620                pointLights[1] = PointLight(0.1f, 0.2f, 1.0f,
3621                    0.5f, 0.5f,
3622                    -5.0f, 5.0f, 30.0f,
3623                    0.17f, 0.01f, 0.0009f);
3624                pointLightCount++;
3625                pointLights[2] = PointLight(0.4f, 0.6f, 1.0f,
3626                    0.5f, 0.5f,
3627                    -60.0f, 5.0f, 100.0f,
3628                    0.17f, 0.01f, 0.0009f);
3629                pointLightCount++;
3630            }
3631            //Luces que se encuentran entre la pecera 5 y el recorrido tubular
3632            if ((camera.getCameraPosition().x >= -100 && camera.getCameraPosition().x <= 100)
3633                && (camera.getCameraPosition().z <= 400 && camera.getCameraPosition().z >= 150))
3634            {
3635                spotLights[0] = SpotLight(0.8f, 0.8f, 0.8f,
3636                    0.5f, 1.0f,
3637                    40.0f, -13.0f, 360.0f,
3638                    -15.0f, -5.0f, -1.0f,
3639                    0.17f, 0.01f, 0.0001f,
3640                    200.0f);
3641                spotLightCount++;
3642
3643                spotLights[1] = SpotLight(0.8f, 0.8f, 0.8f,
3644                    0.5f, 1.0f,
3645                    -60.0f, -13.0f, 360.0f,
3646                    15.0f, -5.0f, -1.0f,
3647                    0.17f, 0.01f, 0.0001f,
3648                    200.0f);
3649                spotLightCount++;
3650
3651                shaderList[0].SetSpotLights(spotLights, 2);
3652                spotLights[0].SetFlash(glm::vec3(-130.0f, -13.0f, 30.0f), glm::vec3(-1.0f, -1.0f, -1.0f));
3653                spotLights[1].SetFlash(glm::vec3(-130.0f, -13.0f, 30.0f), glm::vec3(1.0f, -1.0f, -1.0f));
3654
3655            }
3656
3657            if ((camera.getCameraPosition().x >= -100 && camera.getCameraPosition().x <= 100)
3658                && (camera.getCameraPosition().z <= -300 && camera.getCameraPosition().z >= -400))
3659            {
3660                Sonido = 0;
3661            }
3662
3663        }
```

Como podemos observar en las imágenes anteriormente mostradas, se tiene dividida la iluminación en dos secciones: una sección corresponde a la parte que se encuentra con valores positivos en el eje Y, mientras que la otra sección será aquella en donde el eje Y tenga valores negativos.

Previamente se tenía conocimiento de que, en este caso, el programa solo puede cargar 3 luces del tipo pointlight, y 4 luces del tipo spotlight. Para poder hacer uso de estas luces, principalmente de las luces del tipo puntuales, se implementaron zonas en todo el programa las cuales activarán las luces cuando la cámara se acerque a esa zona. Además de lo anterior dicho, también se tiene que, en ciertos casos, las luces van a cambiar de color según la zona en la que se encuentren.

Audio

Como se vio en el apartado anterior, se tienen zonas en las que se activan las luces correspondientes a esa área. Por lo que, apoyándonos de esas divisiones, podemos utilizar banderas para que en determinadas zonas (por ejemplo, la recepción del acuario) se reproduzca una cierta canción o, en otro caso, se detenga lo que se está escuchando de fondo. Hay que señalar que la librería de audio utilizada se llama IrrKlang, esto debido a su facilidad para configurarlo en el proyecto y usarlo en el mismo. El código implementado se muestra a continuación

```
955     //Musica de espera mientras empieza el programa
956     if (!engine) {
957         printf("No se pudo reproducir el audio");
958     }
959     engine->play2D("Audio/No Chance in Hell.mp3", true);
960     engine->setSoundVolume(0.3);
```

```
1706
1707        engine->stopAllSounds();
1708
```

```
3665        //Musica que se estará usando
3666        switch (Sonido)
3667        {
3668            case 0:
3669                engine->stopAllSounds();
3670                break;
3671            case 1:
3672                if(!engine->isCurrentlyPlaying("Audio/After You ve Gone.mp3")){
3673                    engine->play2D("Audio/After You ve Gone.mp3", true);
3674                    engine->setSoundVolume(0.3);
3675                }
3676                break;
3677            case 2:
3678                if (!engine->isCurrentlyPlaying("Audio/Musica_festival.mp3")) {
3679                    engine->play2D("Audio/Musica_festival.mp3", true);
3680                    engine->setSoundVolume(0.3);
3681                }
3682                break;
3683            case 3:
3684                if (!engine->isCurrentlyPlaying("Audio/Musica_acuario.mp3")) {
3685                    engine->play2D("Audio/Musica_acuario.mp3", true);
3686                    engine->setSoundVolume(0.3);
3687                }
3688                break;
3689            default:
3690                break;
3691        }
```

Animaciones

1. El recorrido de Arturia lily- Realizará un recorrido el cual inicia enfrente del escenario al aire libre, está condicionado por 5 llaves una para marcar el movimiento de Z otra para el movimiento de X, el movimiento de Y para las escaleras, un contador de giros para tener un control de cada movimiento que se realizará y por último un activador el cual da inicio a esta animación (Tecla B) además de contar con movimiento en sus extremidades.

```
//-----------------------------------------------Animacior
//Recorrido de Lily por el festival
if (AvanzaLX && !AvanzaLZ && Cgiros == 0 && ActivadorRL)
{
    if (MAvanzaLX < 80.0f)
    {
        MAvanzaLX += rotLilyOffsetM * deltaTime;
    }
    else
    {
        AvanzaLX = false;
        Cgiros = 1;
    }
}
if (!AvanzaLX && !AvanzaLZ && Cgiros == 1)
{
    if (Mrota < 90.0f)
    {
        Mrota += (rotLilyOffsetM * 2) * deltaTime;
    }
    else
    {
        AvanzaLZ = true;
    }
}
if (!AvanzaLX && AvanzaLZ && Cgiros == 1)
{
    if (MAvanzaLZ < 45.0f)
    {
        MAvanzaLZ += rotLilyOffsetM * deltaTime;
    }
    else
    {
        AvanzaLZ = false;
        Cgiros = 2;
    }
```

```
if (AvanzaLX && !AvanzaLZ && Cgiros == 6)
{
    if (MAvanzaLX > -370.0f)
    {
        MAvanzaLX -= rotLilyOffsetM * deltaTime;
    }
    else
    {
        ActivadorAnimales=true;
        Cgiros = 7;
    }
}

if (AvanzaLX && !AvanzaLZ && Cgiros == 7)
{
    if (MAvanzaLX > -410.0f)
    {
        MAvanzaLX -= rotLilyOffsetM * deltaTime;
        MAvanzaLY -= (rotLilyOffsetM / 3) * deltaTime;
    }
    else
    {
        AvanzaLX = false;
        Cgiros = 8;
    }
}
```

asignación de valores

```
//Arturia Pendragon (Lily)
//Cuerpo
model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(0.0f + MAvanzaLZ, 8.0f + MAvanzaLY, 20.0f + MAvanzaLX));
model = glm::rotate(model, glm::radians(Mrota), glm::vec3(0.0f, 1.0f, 0.0f));
modelaux_cuerpoL = model;
model = glm::scale(model, glm::vec3(10.0f, 10.0f, 10.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Lily_Cuerpo.RenderModel();

//Brazo Izquierdo
model = modelaux_cuerpoL;
model = glm::translate(model, glm::vec3(0.49f * 2, 1.61f * 2, -0.1f * 2));
model = glm::rotate(model, glm::radians(MrotABrazoL), glm::vec3(-1.0f, 0.0f, 0.0f));
modelaux_brazoL = model;
model = glm::scale(model, glm::vec3(10.0f, 10.0f, 10.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Lily_AnteBrazo.RenderModel();
```

2. Las 2 peceras principales con Peces tropicales nadando de un extremo al otro- Principalmente tiene banderas para cada movimiento, el senoidal y el desplazamiento en Z cuenta con un activador el cual se activa cuando lily esté bajando las escaleras del acuario, En el movimiento *Sen* para simular que se mueve de arriba a abajo, mientras que se desplazan 50 unidades para realizar un movimiento más realista, una vez llegando a las 50 unidades rota 180 grados para hacer un desplazamiento ahora hacia el lado contrario.

```
//acuario
//peces
if (rotPecesSen && ActivadorAnimales)
{
    if (MpecesSen < 50.0f)
    {
        MpecesSen += rotpecesSenOffset / 2 * deltaTime;
    }
    else
    {
        rotPecesSen = false;
    }
}
if (!rotPecesSen && ActivadorAnimales)
{
    if (MpecesSen > 0.0f)
    {
        MpecesSen -= rotpecesSenOffset / 2 * deltaTime;
    }
    else
    {
        rotPecesSen = true;
    }
}
```

```
if (rotPeces && MpecesB && ActivadorAnimales)
{
    if (Mpeces < 50.0f)
    {
        Mpeces += MpecesOffset / 2 * deltaTime;
    }
    else
    {
        rotPeces = false;
    }
}

if (MpecesB && !rotPeces)
{
    if (MrotPeces < 180.0f)
    {
        MrotPeces += MpecesOffset * (5) / 2 * deltaTime;
    }
    else
    {
        MpecesB = false;
    }
}

if (!rotPeces && !MpecesB)
{
    if (Mpeces > 0.0f)
    {
        Mpeces -= MpecesOffset / 2 * deltaTime;
    }
    else
    {
        rotPeces = true;
    }
}
```

asignación de valores

```
//animales
//.....pecera entrada 1-----------------------------------------------------------//
model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
//model = glm::translate(model, glm::vec3(0.0f + MAvanzaLZ, 2.7f, 20.0f + MAvanzaLX));
model = glm::translate(model, glm::vec3(30.0f, -20.0f - 2 * sin(glm::radians(MpecesSen)), -290.0f + Mpeces));
model = glm::rotate(model, glm::radians(MrotPeces), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
G_pece_1.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
//model = glm::translate(model, glm::vec3(0.0f + MAvanzaLZ, 2.7f, 20.0f + MAvanzaLX));
model = glm::translate(model, glm::vec3(60.0f, -15.0f - 2 * sin(glm::radians(MpecesSen)), -304.0f + Mpeces));
model = glm::rotate(model, glm::radians(MrotPeces), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
G_pece_1.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
//model = glm::translate(model, glm::vec3(0.0f + MAvanzaLZ, 2.7f, 20.0f + MAvanzaLX));
model = glm::translate(model, glm::vec3(30.0f, -15.0f + 2 * sin(glm::radians(MpecesSen)), -283.0f + Mpeces));
model = glm::rotate(model, glm::radians(MrotPeces), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
G_pece_1.RenderModel();
```

3. La Pecera de la entrada cuenta con Tiburones y un par de peces tropicales avanzan nadando hasta un límite y hace un ciclo- cuenta con 2 llaves un

desplazamiento y una función senoidal de igual forma el tiburón grande cuenta con una animación de cada una de las piezas fragmentadas Aletas y coleta, Su movimiento senoidal va acompañado de un desplazamiento que regresa a su estado inicial para hacer un ciclo continuo en la animación.

```
if (MTiburonSenB && ActivadorAnimales)
{
    if (MTiburonSen < 80.0f)
    {
        MTiburonSen += rotTiburonSenOffset / 2 * deltaTime;
    }
    else
    {
        MTiburonSenB = false;
    }
}
if (!MTiburonSenB && ActivadorAnimales)
{
    if (MTiburonSen > 0.0f)
    {
        MTiburonSen -= rotTiburonSenOffset / 2 * deltaTime;
    }
    else
    {
        MTiburonSenB = true;
    }
}

if (MTiburonB && ActivadorAnimales)
{
    if (MTiburon < 50.0f)
    {
        MTiburon += MdelfinOffset / 2 * deltaTime;
    }
    else
    {
        MTiburon = 0.0;
    }
```

asignación de valores

```
model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(140.0f, -15.0f - 2 * sin(glm::radians(MdelfinSen)), -360.0f + MTiburon));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Tiburon_2.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(140.0f, -15.0f + 2 * sin(glm::radians(MdelfinSen)), -300.0f + MTiburon));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
modelaux_cuerpoTib = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Tiburon_cuerpo.RenderModel();

//aleta der
model = modelaux_cuerpoTib;
model = glm::translate(model, glm::vec3(-4.0f, -5.7f, 45.0f));
model = glm::rotate(model, MEXR * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Tiburon_der_aleta_1.RenderModel();
//
model = modelaux_cuerpoTib;
model = glm::translate(model, glm::vec3(7.0f, -2.0f, 22.0f));
model = glm::rotate(model, MEXR * toRadians, glm::vec3(0.0f, 0.0f, -1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Tiburon_der_aleta_2.RenderModel();
//aleta izq
model = modelaux_cuerpoTib;
model = glm::translate(model, glm::vec3(-11.0f, -6.0f, 44.0f));
model = glm::rotate(model, MEXR * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Tiburon_izq_aleta_1.RenderModel();
```

4. La pecera cilindrica cuenta con caballitos de mar y peces tropicales nadan alrededor del coral- cada grupo de caballitos cuenta con un desplazamiento de 15 unidades y tenemos 2 grupos separados por 15 unidades por lado de la pecera, siendo 4 lados de desplazamiento formando un cuadro, una vez llegando 15 unidades regresa el punto inicial, sin embargo se ve un ciclo completo ya que simulan un bucle sin fin, lo peces en esta pecera son más pequeños y siguen el movimiento de los caballitos de mar,

```
if (MCaballitosSenB && ActivadorAnimales)
{
    if (MCaballitosSen < 80.0f)
    {
        MCaballitosSen += rotCaballitosSenOffset / 2 * deltaTime;
    }
    else
    {
        MCaballitosSenB = false;
    }
}
if (!MCaballitosSenB && ActivadorAnimales)
{
    if (MCaballitosSen > 0.0f)
    {
        MCaballitosSen -= rotCaballitosSenOffset / 2 * deltaTime;
    }
    else
    {
        MCaballitosSenB = true;
    }
}

if (MCaballitosB && ActivadorAnimales)
{
    if (MCaballitos < 15.0f)
    {
        MCaballitos += MCaballitosOffset / 2 * deltaTime;
    }
    else
    {
        MCaballitos = 0.0;
    }
}
```

asignación de valores

```
model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(0.0f + 3.0f, -25.0f + 2 * sin(glm::radians(MCaballitosSen)), -40.0f - MCaballitos));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Caballito.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(2.0f + 3.0f, -20.0f + 2 * sin(glm::radians(MCaballitosSen)), -44.0f - MCaballitos));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Caballito.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(0.0f + 3.0f, -25.0f - 2 * sin(glm::radians(MCaballitosSen)), -44.0f - MCaballitos));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Caballito.RenderModel();
```

5. La pecera del túnel cuenta con medusas y peces tropicales avanzan de extremo a extremo- las medusas están en línea mientras que un grupo avanza el otro retrocede mantienen un desplazamiento menor con una ondulación diferente de su función senoidal.

```
//mmedusas
if (MmedusasSenB && ActivadorAnimales)
{
    if (MmedusasSen < 100.0f)
    {
        MmedusasSen += rotmedusasSenOffset / 2 * deltaTime;
    }
    else
    {
        MmedusasSenB = false;
    }
}
if (!MmedusasSenB && ActivadorAnimales)
{
    if (MmedusasSen > 0.0f)
    {
        MmedusasSen -= rotmedusasSenOffset / 2 * deltaTime;
    }
    else
    {
        MmedusasSenB = true;
    }
}

if (MmedusasB && ActivadorAnimales)
{
    if (Mmedusas < 50.0f)
    {
        Mmedusas += MmedusasOffset / 2 * deltaTime;
    }
    else
    {
        MmedusasB = false;
        Rmedusas = 180;
    }
}
if (!MmedusasB && ActivadorAnimales)
```

asignación de valores

```
model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(-30.0f, -10.0f + 2 * sin(glm::radians(MmedusasSen)), 110.0f + Mmedusas));
model = glm::rotate(model, glm::radians(Rmedusas), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Medusas.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(-30.0f, -10.0f + 2 * sin(glm::radians(MmedusasSen)), 100.0f + Mmedusas));
model = glm::rotate(model, glm::radians(Rmedusas), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Medusas.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(-30.0f, -10.0f + 2 * sin(glm::radians(MmedusasSen)), 90.0f + Mmedusas));
model = glm::rotate(model, glm::radians(Rmedusas), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.1f, 0.1f, 0.1f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Medusas.RenderModel();
```

6. La pecera de los delfines y ballena acompañado de peces tropicales suben a la pecera para poder nadar de un extremo al otro de la pecera y descender-

cuando los delfines llegan a su punto máx de altura empiezan a desplazarse hacia enfrente junto con su movimiento senoidal en Y llegando a 50 unidades y empieza a descender de igual forma cambian su rotación para poder hacer un movimiento más realista además de contar un un movimiento en las extremidades de la ballena.

```
if (MdelfinSenB && ActivadorAnimales)
{
    if (MdelfinSen < 50.0f)
    {
        MdelfinSen += rotdelfinSenOffset / 2 * deltaTime;
    }
    else
    {
        MdelfinSenB = false;
    }
}
if (!MdelfinSenB && ActivadorAnimales)
{
    if (MdelfinSen > 0.0f)
    {
        MdelfinSen -= rotdelfinSenOffset / 2 * deltaTime;
    }
    else
    {
        MdelfinSenB = true;
    }
}
//arriba avance y abajo

if (MdelfinBY && MdelfinB && ActivadorAnimales)
{
    if (MdelfinY < 40.0f)
    {
        Rdelfin = 50;
        MdelfinY += MdelfinOffset / 2 * deltaTime;
    }
    else
    {
        MdelfinBY = false;
    }
}
```

asignación de valores

```
//_--------------------------------Pecera 3-------------------------------------------------------------------//

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(120.0f, -65.0f + 2 * sin(glm::radians(MdelfinSen)) + MdelfinY, 45.0f + MdelfinZ));
model = glm::rotate(model, glm::radians(Rdelfin / 2), glm::vec3(-1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
modelaux_cuerpoBall=model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Ballena.RenderModel();

//aleta der
model = modelaux_cuerpoBall;
model = glm::translate(model, glm::vec3(-50.0f, -20.0f, -12.0f));
model = glm::rotate(model, MEXR * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Ballena_aleta_der.RenderModel();
//aleta izq
model = modelaux_cuerpoBall;
model = glm::translate(model, glm::vec3(50.0f, -20.0f, -12.0f));
model = glm::rotate(model, MEXR * toRadians, glm::vec3(0.0f, 0.0f, -1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Ballena_aleta_izq.RenderModel();


model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(90.0f, -47.0f - 2 * sin(glm::radians(MdelfinSen)) + MdelfinY, 15.0f + MdelfinZ));
model = glm::rotate(model, glm::radians(Rdelfin), glm::vec3(-1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Delfin.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(80.0f, -57.0f + 2 * sin(glm::radians(MdelfinSen)) + MdelfinY, 5.0f + MdelfinZ));
model = glm::rotate(model, glm::radians(Rdelfin), glm::vec3(-1.0f, 0.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.2f, 0.2f, 0.2f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Delfin.RenderModel();
```

7. El hábitat de los pingüinos hacen un show deslizándose por su hábitat- los pingüinos mueven sus aletas además de que dos pingüinos se desplazan uno arriba y otro abajo para simular un deslizamiento mientras que dos pingüinos esperan en los punto de partida de cada uno para hacer un ciclo.

```
//pinguinos
if (MPinguinosSenB && ActivadorAnimales)
{
    if (MPinguinosSen < 15.0f)
    {
        MPinguinosSen += rotPinguinosSenOffset / 2 * deltaTime;
    }
    else
    {
        MPinguinosSenB = false;
    }
}

if (!MPinguinosSenB && ActivadorAnimales)
{
    if (MPinguinosSen > -15.0f)
    {
        MPinguinosSen -= rotPinguinosSenOffset / 2 * deltaTime;
    }
    else
    {
        MPinguinosSenB = true;
    }
}

if (MPinguinosB && ActivadorAnimales)
{
    if (MPinguinos < 10.0f)
    {
        MPinguinos += MPinguinosOffset / 2 * deltaTime;
    }
    else
    {
        MPinguinos = 0.0;
    }
}
```

asignación de valores

```
//cuerpo
model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(-82.0f, -28.0f, 110.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
modelaux_cuerpoPIN1 = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Pinguino_E_aleta_cuerpo.RenderModel();
//aleta der
model = modelaux_cuerpoPIN1;
model = glm::translate(model, glm::vec3(0.8f, 1.2f, 0.0f));
model = glm::rotate(model, MEXR * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Pinguino_E_aleta_der.RenderModel();
//aleta izq
model = modelaux_cuerpoPIN1;
model = glm::translate(model, glm::vec3(-0.8f, 1.2f, 0.0f));
model = glm::rotate(model, MEXR * toRadians, glm::vec3(0.0f, 0.0f, -1.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Pinguino_E_aleta_izq.RenderModel();

//cuerpo
model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(-82.0f, -28.0f, 115.0f));
model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
modelaux_cuerpoPIN2 = model;
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Pinguino_E_aleta_cuerpo.RenderModel();
```

8. El espectáculo de los delfines saltan alrededor de la isla- cada delfín tiene una posición inicial de la cual se desplaza 50 unidades en X mientras que se desplaza 25 unidades arriba en y y llegando a los 25 en X empieza a bajar en Y para hacer un bucle con los cuatro delfines en cada extremo de la isla

```
//movimiento del salto
if (MdelfinBS && ActivadorAnimales)
{
    if (MdelfinZS < 50.0f)
    {
        MdelfinZS += MdelfinOffset / 2 * deltaTime;
    }
    else
    {
        MdelfinZS = 0.0f;
    }
}

//salto
if (MdelfinBsalto && ActivadorAnimales)
{
    if (MdelfinSalto < 25.0f)
    {
        MdelfinSalto += MdelfinOffset / 2 * deltaTime;
    }
    else
    {
        MdelfinBsalto = false;
    }
}

if (!MdelfinBsalto && ActivadorAnimales)
{
    if (MdelfinSalto > 0.0f)
    {
        MdelfinSalto -= MdelfinOffset / 2 * deltaTime;
    }
    else
    {
```

asignación de valores

```
model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(10.0f - MdelfinZS, -56.0f + MdelfinSalto, 290.0f));
model = glm::rotate(model, 90 * toRadians, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.15f, 0.15f, 0.15f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Delfin.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(-40.0f, -56.0f + MdelfinSalto, 290.0f + MdelfinZS));
model = glm::scale(model, glm::vec3(0.15f, 0.15f, 0.15f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Delfin.RenderModel();

model = glm::mat4(1.0);
color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(-40.0f + MdelfinZS, -56.0f + MdelfinSalto, 340.0f));
model = glm::rotate(model, -90 * toRadians, glm::vec3(0.0f, -1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.15f, 0.15f, 0.15f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Delfin.RenderModel();
```

9. Keyframes con padoru bailando

Animaciones de Keyframe Para las animaciones de keyframe se usaron diferentes variables ya que teníamos que usar dos keyframes diferentes por lo que usamos un KeyFrameN (para Nero_Paforu) KeyFrameK (para Koshiro) y cada una tiene keyframe diferentes mientras padoru tiene 10 koshiro tiene 25 y cada una realiza movimientos diferentes

```
//-----------------------------------
//Nero
KeyFrameN[0].movNero_x = 0.0f;
KeyFrameN[0].movNero_y = 0.0f;
KeyFrameN[0].movNero_z = 0.0f;
KeyFrameN[0].giroNero = 0.0f;


KeyFrameN[1].movNero_x = 1.0f;
KeyFrameN[1].movNero_y = 0.0f;
KeyFrameN[1].movNero_z = 2.0f;
KeyFrameN[1].giroNero = 0.0f;


KeyFrameN[2].movNero_x = 2.0f;
KeyFrameN[2].movNero_y = 0.0f;
KeyFrameN[2].movNero_z = 0.0f;
KeyFrameN[2].giroNero = 0.0f;


KeyFrameN[3].movNero_x = 3.0f;
KeyFrameN[3].movNero_y = 0.0f;
KeyFrameN[3].movNero_z = -2.0f;
KeyFrameN[3].giroNero = -180.0f;

KeyFrameN[4].movNero_x = 3.0f;
KeyFrameN[4].movNero_y = 2.0f;
KeyFrameN[4].movNero_z = 0.0f;
KeyFrameN[4].giroNero = -180;
```

asignación de valores

Extras

Movimientos de Nero para animación “Cantar” Cuenta con banderas para activar su movimiento y un activador el cual activa su animación

```
//Animación de Nero
//Movimiento en el Antebrazo
if (rotABN && ActivadorN)
{
    if (MrotABrazoN < 50.0f)
    {
        MrotABrazoN += rotNeroOffset * deltaTime;
    }
    else
    {
        rotABN = false;
    }
}
if (!rotABN && ActivadorN)
{
    if (MrotABrazoN > -10.0f)
    {
        MrotABrazoN -= rotNeroOffset * deltaTime;
    }
    else
    {
        rotABN = true;
    }
}
//Movimiento en el Brazo
```

asignación de valores

```
//Nero
model = glm::mat4(1.0);
model = glm::translate(model, glm::vec3(0.0f * 2, (5.0f + 4.5f) * 2, -10.0f));
modelaux_cuerpoN = model;
model = glm::scale(model, glm::vec3(2.0f * 2, 2.0f * 2, 2.0f * 2));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Nero_Cuerpo.RenderModel();
//Micro
model = modelaux_cuerpoN;
model = glm::translate(model, glm::vec3(0.0f * 2, -4.0f * 2, 1.0f * 2));
model = glm::scale(model, glm::vec3(0.2f * 2, 0.2f * 2, 0.2f * 2));
//model = glm::rotate(model, 135 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Micro.RenderModel();
//brazo der
model = modelaux_cuerpoN;
model = glm::translate(model, glm::vec3(0.65f * 2, 0.35f * 2, -0.02f * 2));
model = glm::rotate(model, -30 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians((MrotABrazoN)), glm::vec3(0.0f, -1.0f, -1.0f));
modelaux_brazoN = model;
model = glm::scale(model, glm::vec3(2.0f * 2, 2.0f * 2, 2.0f * 2));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Nero_AnteBrazo.RenderModel();
```

Movimientos de Koshiro para animación “Tocar guitarra” Cuenta con banderas para activar su movimiento y un activador el cual activa su animación

```
//Animación Koshiro
//Movimiento en el Antebrazo
if (rotABKS && ActivadorKS)
{
    if (MrotABrazoKS < 30.0f)
    {
        MrotABrazoKS += rotKoshiroOffset * deltaTime;
    }
    else
    {
        rotABKS = false;
    }
}
if (!rotABKS && ActivadorKS)
{
    if (MrotABrazoKS > -20.0f)
    {
        MrotABrazoKS -= rotKoshiroOffset * deltaTime;
    }
    else
    {
        rotABKS = true;
    }
}
```

asignación de valores

```
//Koshiro
model = glm::mat4(1.0);
//poskoshiro = glm::vec3(posXkoshiro + movkoshiro_x, posYkoshiro + movkoshiro_y, poszkoshiro + movkoshiro_z);
//model = glm::translate(model, poskoshiro);
model = glm::translate(model, glm::vec3((-4.0f * 2) + movkoshiro_x, ((5.0f + 3.55f) * 2) + movkoshiro_y, (-20.0f ) + movkoshiro_z));
model = glm::rotate(model, girokoshiro * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
modelaux_cuerpoKS = model;
model = glm::scale(model, glm::vec3(5.0f * 2, 5.0f * 2, 5.0f * 2));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
koshiro_Cuerpo.RenderModel();

//Guitarra
model = modelaux_cuerpoKS;
model = glm::translate(model, glm::vec3(0.8f * 2, -1.3f * 2, 0.7f * 2));
model = glm::scale(model, glm::vec3(0.4f * 2, 0.4f * 2, 0.4f * 2));
model = glm::rotate(model, 15 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, -35 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Guitarra.RenderModel();

//Brazo Izquierdo
model = modelaux_cuerpoKS;
model = glm::translate(model, glm::vec3(0.47f * 2, 0.01f * 2, -0.05f * 2));
model = glm::rotate(model, -45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
modelaux_brazoKS = model;
model = glm::scale(model, glm::vec3(5.0f * 2, 5.0f * 2, 5.0f * 2));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
koshiro_AnteBrazo.RenderModel();
```

Movimientos de Koharu para animación “Tocar batería” Cuenta con banderas para activar su movimiento y un activador el cual activa su animación

```
//Animación de Koharu
//Movmiento en el Antebrazo
if (rotABK && ActivadorK)
{
    if (MrotABrazoK < 50.0f)
    {
        MrotABrazoK += rotKoharuOffset * deltaTime;
    }
    else
    {
        rotABK = false;
    }
}
if (!rotABK && ActivadorK)
{
    if (MrotABrazoK > -40.0f)
    {
        MrotABrazoK -= rotKoharuOffset * deltaTime;
    }
    else
    {
        rotABK = true;
    }
```

asignación de valores

```
//koharu
//Cuerpo
model = glm::mat4(1.0);
//color = glm::vec3(0.0f, 0.0f, 0.3f);
model = glm::translate(model, glm::vec3(4.0f * 2, (5.0f + 3.0f) * 2, -20.0f));
modelaux_cuerpoK = model;
model = glm::scale(model, glm::vec3(5.0f * 2, 5.0f * 2, 5.0f * 2));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Koharu_Cuerpo.RenderModel();
//Bateria
model = modelaux_cuerpoK;
model = glm::translate(model, glm::vec3(2.0f * 2, -3.0f * 2, 5.0f * 2));
model = glm::scale(model, glm::vec3(0.15f * 2, 0.15f * 2, 0.15f * 2));
//model = glm::rotate(model, 135 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Bateria.RenderModel();
//Silla humilde
model = modelaux_cuerpoK;
model = glm::translate(model, glm::vec3(0.0f * 2, -4.5f * 2, -0.5f * 2));
model = glm::scale(model, glm::vec3(0.09f * 2, 0.09f * 2, 0.09f * 2));
//model = glm::rotate(model, 135 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Silla_R.RenderModel();

//Brazo Izquierdo
model = modelaux_cuerpoK;
model = glm::translate(model, glm::vec3(0.43f * 2, -0.15f * 2, -0.05f * 2));
model = glm::rotate(model, -45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
model = glm::rotate(model, -60 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::rotate(model, glm::radians((MrotABrazoK)), glm::vec3(0.0f, 0.0f, 1.0f));
modelaux_brazoK = model;
model = glm::scale(model, glm::vec3(5.0f * 2, 5.0f * 2, 5.0f * 2));
glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
Koharu_AnteBrazo.RenderModel();
```

carga de modelos

```
////Artoria Pendragon (Lily)
Lily_AnteBrazo = Model();
Lily_AnteBrazo.LoadModel("Models/Lily/Lily_AnteBrazo_1.obj");
Lily_AnteBrazo_2 = Model();
Lily_AnteBrazo_2.LoadModel("Models/Lily/Lily_AnteBrazo_2.obj");
Lily_Brazo = Model();
Lily_Brazo.LoadModel("Models/Lily/Lily_Brazo_1.obj");
Lily_Brazo_2 = Model();
Lily_Brazo_2.LoadModel("Models/Lily/Lily_Brazo_2.obj");
Lily_Pierna = Model();
Lily_Pierna.LoadModel("Models/Lily/Lily_Pierna_1.obj");
Lily_Pierna_2 = Model();
Lily_Pierna_2.LoadModel("Models/Lily/Lily_Pierna_2.obj");
Lily_Pie = Model();
Lily_Pie.LoadModel("Models/Lily/LilY_Pie_1.obj");
Lily_Pie_2 = Model();
Lily_Pie_2.LoadModel("Models/Lily/Lily_Pie_2.obj");
Lily_Cabeza = Model();
Lily_Cabeza.LoadModel("Models/Lily/Lily_Cabeza.obj");
Lily_Cuerpo = Model();
Lily_Cuerpo.LoadModel("Models/Lily/Lily_Cuerpo.obj");


////nero
Nero_Cuerpo = Model();
Nero_Cuerpo.LoadModel("Models/Nero/nero_cuerpo.obj");
Nero_AnteBrazo = Model();
Nero_AnteBrazo.LoadModel("Models/Nero/nero_antebrazo_1.obj");
Nero_AnteBrazo_2 = Model();
Nero_AnteBrazo_2.LoadModel("Models/Nero/nero_antebrazo_2.obj");
Nero_Brazo = Model();
Nero_Brazo.LoadModel("Models/Nero/nero_brazo_1.obj");
Nero_Brazo_2 = Model();
Nero_Brazo_2.LoadModel("Models/Nero/nero_brazo_2.obj");

////koshiro
koshiro_AnteBrazo = Model();
koshiro_AnteBrazo.LoadModel("Models/koshiro/Koshiro_ABrazo_1.obj");
koshiro_AnteBrazo_2 = Model();
koshiro_AnteBrazo_2.LoadModel("Models/koshiro/Koshiro_ABrazo_2.obj");
koshiro_Brazo = Model();
koshiro_Brazo.LoadModel("Models/koshiro/Koshiro_Brazo_1.obj");
koshiro_Brazo_2 = Model();
koshiro_Brazo_2.LoadModel("Models/koshiro/Koshiro_Brazo_2.obj");
koshiro_Pierna = Model();
koshiro_Pierna.LoadModel("Models/koshiro/Koshiro_Pierna_1.obj");
koshiro_Pierna_2 = Model();
koshiro_Pierna_2.LoadModel("Models/koshiro/Koshiro_Pierna_2.obj");
koshiro_Pie = Model();
koshiro_Pie.LoadModel("Models/koshiro/Koshiro_Pie_1.obj");
koshiro_Pie_2 = Model();
koshiro_Pie_2.LoadModel("Models/koshiro/Koshiro_Pie_2.obj");
koshiro_Cuerpo = Model();
koshiro_Cuerpo.LoadModel("Models/koshiro/Koshiro_Cuerpo.obj");
```

```
////Koharu
Koharu_AnteBrazo = Model();
Koharu_AnteBrazo.LoadModel("Models/Koharu/Koharu_AnteBrazo_1.obj");
Koharu_AnteBrazo_2 = Model();
Koharu_AnteBrazo_2.LoadModel("Models/Koharu/Koharu_AnteBrazo_2.obj");
Koharu_Brazo = Model();
Koharu_Brazo.LoadModel("Models/Koharu/Koharu_Brazo_1.obj");
Koharu_Brazo_2 = Model();
Koharu_Brazo_2.LoadModel("Models/Koharu/Koharu_Brazo_2.obj");
Koharu_Pierna = Model();
Koharu_Pierna.LoadModel("Models/Koharu/Koharu_Pierna_1.obj");
Koharu_Pierna_2 = Model();
Koharu_Pierna_2.LoadModel("Models/Koharu/Koharu_Pierna_2.obj");
Koharu_Pie = Model();
Koharu_Pie.LoadModel("Models/Koharu/Koharu_Pie_1.obj");
Koharu_Pie_2 = Model();
Koharu_Pie_2.LoadModel("Models/Koharu/Koharu_Pie_2.obj");
Koharu_Cuerpo = Model();
Koharu_Cuerpo.LoadModel("Models/Koharu/Koharu_Cuerpo.obj");

////padoru
Padoru = Model();
Padoru.LoadModel("Models/PAdoru/Nero_Padoru.obj");


//animales acuario
G_pece_1 = Model();
G_pece_1.LoadModel("Models/G_Pez1/Peces1.obj");
G_pece_2 = Model();
G_pece_2.LoadModel("Models/G_Pez2/Peces2.obj");
G_pece_3 = Model();
G_pece_3.LoadModel("Models/G_Pez3/Peces3.obj");
Tiburon_cuerpo = Model();
Tiburon_cuerpo.LoadModel("Models/Tiburon_1/Tiburonsin_uhaha.obj");
Tiburon_der_aleta_1 = Model();
Tiburon_der_aleta_1.LoadModel("Models/Tiburon_1/aletas_der_1.obj");
Tiburon_der_aleta_2 = Model();
Tiburon_der_aleta_2.LoadModel("Models/Tiburon_1/aletas_der_2.obj");
Tiburon_izq_aleta_1 = Model();
Tiburon_izq_aleta_1.LoadModel("Models/Tiburon_1/aletas_izq_1.obj");
Tiburon_izq_aleta_2 = Model();
Tiburon_izq_aleta_2.LoadModel("Models/Tiburon_1/aletas_izq_2.obj");
Tiburon_cola_1 = Model();
Tiburon_cola_1.LoadModel("Models/Tiburon_1/coleta_1.obj");
Tiburon_cola_2 = Model();
Tiburon_cola_2.LoadModel("Models/Tiburon_1/coleta_2.obj");
Tiburon_2 = Model();
Tiburon_2.LoadModel("Models/Tiburon_2/Tiburon.obj");
Medusas = Model();
Medusas.LoadModel("Models/medusas/medusas.obj");
Delfin = Model();
Delfin.LoadModel("Models/delfin/delfin.obj");
Ballena = Model();
Ballena.LoadModel("Models/Ballena/ballena.obj");
Ballena_aleta_der = Model();
Ballena_aleta_der.LoadModel("Models/Ballena/aletas_der.obj");
```

```
Ballena_aleta_izq = Model();
Ballena_aleta_izq.LoadModel("Models/Ballena/aletas_izq.obj");
Ballena_coleta = Model();
Ballena_coleta.LoadModel("Models/Ballena/coleta.obj");
Caballito = Model();
Caballito.LoadModel("Models/Caballito/caballito.obj");
Pinguino_E = Model();
Pinguino_E.LoadModel("Models/Pingus/grapem1.obj");
Pinguino_E_aleta_der = Model();
Pinguino_E_aleta_der.LoadModel("Models/Pingus/Pengu_Aleta_der.obj");
Pinguino_E_aleta_izq = Model();
Pinguino_E_aleta_izq.LoadModel("Models/Pingus/Pengu_Aleta_izq.obj");
Pinguino_E_aleta_cuerpo = Model();
Pinguino_E_aleta_cuerpo.LoadModel("Models/Pingus/Pengu_cuerpo.obj");

//extras
Lucy = Model();
Lucy.LoadModel("Models/Lucy/Lucy.obj");
vik = Model();
vik.LoadModel("Models/vik/vik.obj");
elf = Model();
elf.LoadModel("Models/elf/elf.obj");
```